ST.GAME

* P.S. *St.Game*, by the way, is the new name of *Softline*. Same mag, new name. You're the first to know.

PROGRAMS IN PULP

ATARI DUNGEON FUN

LEVELS ONE AND TWO

BY BILL BEYERMANN

It's catch-up time! Here, bringing Atari people up to date on the construction of *St.Game*'s computer dungeon and the action therein, are the Atari versions of levels 1 and 2. They form a rather fascinating lesson in translation—even those who don't have an Atari may be interested in seeing some of the differences between Applesoft and Atari Basic.

Atari Basic requires more work to do certain things. In order to put convenient graphics and sound keywords in their Basic, Atari left out some of the niceties of other Basics, such as string arrays. Also, functions to scroll a line, clear to the end of the line, clear to the end of the frame (screen), and perform the inverse printing of a variable are not available.

The first noticeable difference is the bunch of subroutines between lines 50 and 100 in listing 1. Line 60 scrolls one line (Apple *call – 912*). It does this by positioning the cursor on the last screen line and printing a character. XX is the memory location of where the cursor is being positioned (column 0, row 23—the leftmost position on the bottom line). The character being printed is the one already in that position. The horizontal (H) and vertical (V) positions of the cursor are saved before the scroll and then restored (the same is done in the next two subroutines).

Line 62 clears to the end of the line (Apple *call – 868*) by printing a string of blanks (BL$) that is one less than the remaining columns. The rightmost column (39) is then poked with a blank (*poke 93, 0*). The poke is necessary because the last screen position (39, 23) cannot be printed without causing a scroll. Location 93 stores the character under the cursor. By the way, you may notice that there are only thirty-seven spaces in BL$ in line 5030. The Atari boots up with a thirty-eight-character-wide screen, since some TVs don't properly display the leftmost one or two columns. Leaving well enough alone, we make I$ three spaces instead of five and change the spacing in a few other lines.

Lines 66 through 69 are the "clear to the end of the screen" routine (Apple *call – 958*). The first time the loop is executed, the current line is cleared to the end. H1 is then set to two to clear all subsequent lines completely. (Notice that the Apple displays columns 1 through 40 and rows 1 through 24, while the Atari displays columns 0 through 39 and rows 0 through 23.)

Line 70 is the start of the string array subroutine. The Atari does not handle string arrays but does allow strings of up to 32,767 characters, so a string array can be emulated by putting all the elements in one long string and separating them by delimiters, for which we use commas. Everything from the beginning of the string to the first comma is the zeroth element, from there to the second comma is the first element, et cetera. Using this method, the last element must be followed by a comma.

To see how this works, let's look at line 350. It originally calls for O$(DOING) to be printed. If you look at line 5360, you'll see that O$ is now a string containing three elements followed by commas. To use the subroutine, we first set the dummy string AZ$ to O$ and the dummy IZ to DOING, the element number. AZ$ and IZ are used to pass information to the subroutine. AZ$ also passes the final result back to the main program. When the subroutine is called, the element number (EL) is set to zero, the starting pointer (SP) is set to one, and the ending pointer (LP) is set to one (the first character in the string). The string is then searched for a comma; when it is found, LP is set to the character before the comma. If this is the element we want (EL = IZ), AZ$ is set to the substring of AZ$—AZ$ = AZ$(SP,LP)—from the starting character (SP) to the ending character (LP), and the routine returns. If this isn't the element we want, the element number is increased (EL = EL + 1), the starting pointer is set to the first character of the next element (SP = IX + 1), and line 74 searches for the next comma. IX is set to 999 in line 82 so that the loop will terminate normally. Jumping out of a loop leaves a bunch of junk on the stack, and if you jump out of enough loops with different loop counters, you can overflow the stack.

Line 90 is for inverse printing. The variable to be reversed is sent via AZ$. A numeric variable must first be changed to a string using the STR$ function. The routine simply adds 128 to each character of the string and returns it as AZ$.

Keyboard input requires the keyboard to be opened (line 5020) and the keystrokes to be read (line 380) as ASCII values (numeric instead of string). The *home* command becomes a PRINT CH$(125) or a PRINT "(CLR/HOME)". The buzzer is sounded using ASCII 253. The ASCII values for the right and left arrow keys are 31 and 30, and we've added 42 and 43 (the * and +) so you don't have to hold the control key to get the arrow functions. Quotation marks cannot be used in data statements, since the Atari reads them as part of the data.

Apple's *htab x* and *vtab y* become Atari's *position x – 1, y – 1*: *htab x* becomes *position x – 1, peek(84)*; *vtab y* becomes *position peek(85), y – 1*. *peek(37)* goes to *peek(84)*. *peek(36)* goes to *peek(85)*. A print command followed by more than one item requires ;'s between all the items.

Atari lines are limited to three screen lines, which means that long Apple lines must be broken into two or more lines. All string variables must be dimensioned.

The cursor stays on during printing, so unless you want to see it scooting across the screen, it must be turned off with *poke 752, 1* (or any number except zero). That's right, *poke 752, 0* turns it back on. You can turn it off during initialization, then turn it on just before a get, and off immediately after the set.

Also, the Atari accepts variable names of up to 120 characters, so EN and ENERGY are different variables.

It's recommended that you change the delay timer (TIME) in line 5170 to 30 or less, since the Atari runs slower than the Apple.

The "Dungeon Fun, Part 2" listing contains only the lines that need changing or adding, except for a few remarks. Notice that line 500 is just a reminder to delete lines 500 through 730. We also encounter two more of Atari Basic's limitations; no integer variables (% declarations are not allowed) and no arrays over two dimensions. To handle the three-dimensional array (SQ) introduced in this part, we convert it to a one-dimensional array, which is the way multidimensioned arrays are stored in memory. The array will have the rightmost element varying fastest, so the order of elements is 0,0,0 – 0,0,1 – 0,0,2 – 0,0,3 – 0,1,0 – 0,1,1 – and so on. To choose a particular element, the position value of the first dimension (a in the example below) is multiplied by the number of items in the third dimension (4), times the number of items in the second dimension (3), plus the position value of the second dimension (b), times the number of items in the third dimension (4), plus the position value of the third dimension (c). SQ has three, three, and four items in its first, second, and third dimensions respectively. So SQ(a, b, c) is SQ(12*a + 4*b + c), as in lines 4540 to 4555.

Type in listing 2, after typing *new*, then save it with the *list* command. Then *load* listing 1 and *enter* listing 2, adding the movement lines to the combat program. Lines to be deleted are remarks in listing 2. Since they overwrite the old lines, they cause no problem, except in taking up space. If you wish, you may delete them before saving the new combined program.

That's it. Have fun.

Listing 1

```
3 REM
4 REM THE DUNGEON MACHINE
5 REM LEVEL 1: COMBAT
6 REM V: 67
7 REM
```

```
8 REM
50 GOTO 5000:REM INITIALIZE
60 H = PEEK(85):V = PEEK(84):XX = PEEK(88) + 256*PEEK(89) + 920:POSITION 0,23:? CHR$(PEEK(XX) + 32):POKE
   84,V:POKE 85,H:RETURN
62 H = PEEK(85):V = PEEK(84):? BL$(H - 1);:POKE 93,0:POKE 84,V:POKE 85,H:RETURN
66 H = PEEK(85):V = PEEK(84):H1 = H
67 FOR X = 1 TO 24 - V:? BL$(H1 - 1);:POKE 93,0
68 POSITION 2,PEEK(84) + 1:H1 = 2:NEXT X
69 POKE 84,V:POKE 85,H:RETURN
70 EL = 0:SP = 1:LP = 1
72 FOR IX = 1 TO 999
74 IF AZ$(IX,IX) < > "," THEN 84
76 LP = IX - 1
78 IF EL < > IZ THEN EL = EL + 1:SP = IX + 1:GOTO 84
80 AZ$ = AZ$(SP,LP)
82 IX = 999
84 NEXT IX
86 RETURN
90 FOR IX = 1 TO LEN(AZ$):AZ$(IX,IX) = CHR$(ASC(AZ$(IX,IX)) + 128):NEXT IX:RETURN
100 REM + + + EACH ROUND + + +
110 REM ** NEW PAGE **
115 REM ENTER W/CUR JUST BELOW LAST LINE TO BE RETAINED
120 CV = PEEK(84)
130 GOSUB 66:GOSUB 1240
140 FOR X = 1 TO CV - 19
150 GOSUB 60:NEXT X
160 FOR X = 0 TO 1:ENERGY(X) = ENERGY(X) + 15
170 IF ENERGY(X) > 100 THEN ENERGY(X) = 100
180 NEXT X
200 REM ** PRINT STATUS **
210 POSITION 2,0:? ST$
220 GOSUB 62:PRINT NAME$;
230 POSITION 9,1:? PROF$;
240 POSITION 12,1:? HEALTH(1);
250 POSITION 19,1:? ENERGY(1);
260 POSITION 30,1:? HP;
270 POSITION 35,1:? EP
280 GOSUB 62
300 REM *** PROMPT AND INPUT
310 GOSUB 1240
320 POSITION 2,21:? P$;
330 CH = PEEK(85)
340 ? :? :GOSUB 1240
350 AZ$ = O$:IZ = DOING:GOSUB 70:? AZ$;
360 GOSUB 1240
370 POSITION CH + 1,21:POKE 752,0:? " ";
380 GET #1,Q:POKE 752,1:REM READ INPUT
400 REM ** DECODE & BRANCH **
420 IF Q = 27 THEN END
430 POSITION 2,PEEK(84):GOSUB 66
450 ON DOING GOTO 900,2000:REM
500 REM + + + EXPLORE + + +
510 REM ** STRAIGHT AHEAD **
520 IF Q < > 32 AND Q < > 70 THEN 600:REM TURNS OR ERROR
530 IF NOT WALL THEN 720
550 ? Z$;I$;"(THUMP) OUCH! THAT'S A"
560 ? I$;"WALL, NOT A DOOR!"
570 GOTO 800:REM
600 REM ** RIGHT OR LEFT **
610 IF Q = 31 OR Q = 82 OR Q = 42 THEN DD = + 1:GOTO 640
620 IF Q = 30 OR Q = 76 OR Q = 43 THEN DD = - 1:GOTO 640
630 GOTO 1300:REM BAD INPUT
640 D = D + DD
650 IF D > 3 THEN D = 0
660 IF D < 0 THEN D = 3
670 WALL = (2*INT(D/2) < D)
700 REM ** WHAT YOU SEE **
710 IF WALL THEN ? I$;W$:GOTO 800
720 R = INT(3*RND(5))
730 AZ$ = C$:IZ = R:GOSUB 70:? I$;AZ$
800 REM ** AN ENCOUNTER? **
810 IF SAFE*RND(5) > 1 THEN 110:REM NO ENCOUNTER
```

```
820 DOING = 1
830 ? :? I$;E$
840 GOTO 100:REM
900 REM ** RUN OR FIGHT? **
904 IF Q = 70 THEN 960
906 IF Q < > 82 THEN 1300
910 IF RND(6) < 0.5 THEN 950
920 ? I$;"YOU GOT AWAY, THIS TIME!":? :GOSUB 1230
930 ENERGY(1) = INT(ENERGY(1)/2):D = INT(3*RND(7))
940 DOING = 0:GOTO 670
950 ? I$;"NO GOOD—HE'S TOO CLOSE!"
960 ? :? I$;"HE'S A FIGHTER, ABOUT YOUR"
970 ? I$;"SIZE, AND WITH SIMILAR"
980 ? I$;"EQUIPMENT. GOOD LUCK!"
990 ? :DOING = 2:GOTO 1900:REM
1000 REM + + + UTILITIES + + +
1010 REM ** RANDOM MOD **
1015 REM ENTER WITH RIN PRESET
1020 SIGN = 1 - 2*(RND(7) < 0.5)
1030 RO = INT(RIN*RND(6)^2) + 1
1040 RO = SIGN*RO/2
1050 RETURN
1200 REM ** DELAY TIMER **
1210 GOSUB 1220:REM 16 CYCLES
1220 GOSUB 1230:REM 8 CYCLES
1230 GOSUB 1240:REM 4 CYCLES
1240 GOSUB 1250:REM 2 CYCLES
1250 FOR T = 1 TO TIME
1260 NEXT T:RETURN
1270 AZ$ = DAM$:IZ = HIT( NOT W):GOSUB 70:? AZ$
1300 REM ** BAD INPUT **
1310 POSITION CH + 2,21:? CHR$(Q)
1320 GOSUB 1340:POSITION PEEK(85),19
1330 GOSUB 62:POSITION CH + 2,21:? " ";:GOTO 200
1340 GOSUB 1240:GOSUB 1350
1350 GOSUB 1240:POSITION 2,23
1360 AZ$ = O$:IZ = DOING:GOSUB 70:? Z$;AZ$;:POSITION 2,PEEK(84)
1370 GOSUB 1230:GOSUB 62
1380 RETURN
1900 REM ** NEW OPPONENT **
1910 HEALTH(0) = 100
1920 ENERGY(0) = 100
1930 AC(0) = 100
1940 DC(0) = 100
1950 DEAD = 0
1960 GOTO 100
1990 REM
2000 REM + + + FIGHT! + + +
2010 REM ** INPUTS **
2020 IN(0) = INT(RND(5)*SQR(1.3*ENERGY(0))) + 1
2025 IF IN(0) > 9 THEN IN(0) = 9
2030 IN(1) = Q - 48
2040 IF Q < 49 OR Q > 57 THEN 3000
2050 IF IN(1)^2*2/3 < ENERGY(1) THEN 2100
2060 ? I$;Z$;"YOU'RE TOO TIRED";
2070 GOSUB 1220:POSITION 2,PEEK(84)
2080 GOSUB 62:GOTO 200
2100 REM ** THE ATTACKS **
2103 ? I$;"YOUR INPUT:";
2105 ? IN(1);" HIS:";IN(0);
2110 GOSUB 2500:REM SETUP
2113 ? " HIS EN:";ENERGY(0)
2115 ? I$;"YR AK:";AI(1);" DF:";DI(1);
2117 ? " HIS AK:";AI(0);" DF:";DI(0)
2130 GOSUB 2600:REM ATTACK AND DEFENSE
2140 GOSUB 2800:REM DAMAGE
2150 GOSUB 2900:REM REPORT
2180 IF  NOT DEAD THEN 100
2200 REM ** IN CASE OF DEATH **
2210 ? CHR$(125):POSITION 9,9
2220 IF HEALTH(1) < 1 THEN ? "OOPS! TOO BAD....":END
2230 ? "CONGRATULATIONS!"
```

```
2240 POSITION 11,13
2250 ? "YOU GOT HIM!"
2260 GOSUB 1220:DOING = 0
2270 ? :? :?
2280 IF 3*RND(5) < 1 THEN 100:REM NO LOOT
2300 REM ** LOOT! **
2310 ? I$;"AND HE WAS CARRYING"
2320 ? :GOSUB 1230
2330 LT = INT(1.7*RND(5))
2340 LQ = INT(5*RND(5)) + 2
2350 IF LT THEN HP = HP + LQ
2360 IF  NOT LT THEN EP = EP + LQ
2370 AZ$ = LT$:IZ = LT:GOSUB 70:? I$;I$;LQ;" ";AZ$
2380 GOSUB 1220:GOTO 100
2500 REM ** SET UP ATTACKS **
2510 W = 0:GOSUB 2530
2520 W =  NOT W
2530 VIT(W) = INT((ENERGY(W) + HEALTH(W))/2)
2540 ENERGY(W) = ENERGY(W) - INT(IN(W)^2*2/3)
2550 IN = IN(W) - 1
2560 AI(W) = INT(IN/3)
2570 DI(W) = IN - 3*AI(W)
2590 RETURN :REM
2600 REM ** COMPUTE ATTACKS **
2610 GOSUB 2630
2620 W =  NOT W
2630 AI = (AI(W) + 3)/4:IF AI < 1 THEN AI = 0:ENERGY(W) =  ENERGY(W) + 7
2640 DI = (DI( NOT W) + 3)/4
2643 WHO$ = "HIS":IF W THEN WHO$ = "YOUR"
2645 ? I$;WHO$;" ATK ";AI;
2647 ? "  AGAINST ";DI
2650 RIN = AI*VIT(W)*AC(W)/100
2655 ? I$;" RIN = ";RIN;
2660 GOSUB 1000:RO = RIN + RO
2665 ? " RO = ";RO;
2670 HIT = RO - DI*VIT( NOT W)*DC( NOT W)/100
2680 ? ": HIT = ";HIT;" ";
2710 IF HIT < 1 THEN HIT(W) = 0:GOTO 2750
2720 IF HIT < 15 THEN HIT(W) = 1:GOTO 2750
2730 IF HIT < 50 THEN HIT(W) = 2:GOTO 2750
2740 HIT(W) = 4
2750 REM
2755 AZ$ = STR$(HIT(W)) :GOSUB 90:? AZ$
2760 RETURN :REM
2800 REM ** DAMAGE **
2810 GOSUB 2830
2820 W =  NOT W
2830 DAM = HIT(W)*INT(10*RND(5) + 1) + 10*(HIT(W) - 1)
2840 IF DAM < 0 THEN DAM = 0
2850 HEALTH( NOT W) = HEALTH( NOT W) - DAM
2860 IF HEALTH( NOT W) < 1 THEN DEAD = 1
2870 RETURN :REM
2900 REM ** REPORT **
2910 W = 0
2920 GOSUB 2940
2930 W =  NOT W
2940 ? :? I$;
2950 AZ$ = HIT$:IZ = W:GOSUB 70:IF  NOT HIT( NOT W) THEN ? AZ$(1,4);:GOTO 2970
2960 AZ$ = HIT$:IZ = W:GOSUB 70:? AZ$;
2970 AZ$ = DAM$:IZ = HIT( NOT W):GOSUB 70:? AZ$
2990 RETURN :REM
3000 REM ** HEAL AND ENERGIZE **
3010 IF Q = 81 THEN DOING = 0:? I$;"COWARD!":GOTO 100
3020 IF Q < > 69 and Q < > 72 THEN 1300
3030 FIX = HEALTH(1):USE = HP
3040 IF Q = 69 THEN FIX = ENERGY(1):USE = EP
3050 IF USE = 0 THEN 3130
3060 USE = USE - 1
3070 FIX = FIX + 30 + INT(40*RND(5)):IF FIX > 100 THEN FIX = 100
3080 IF Q = 69 THEN ENERGY(1) = FIX:EP = USE
3090 IF Q = 72 THEN HEALTH(1) = FIX:HP = USE
3100 IN(1) = 3:GOTO 2100
```

```
3130 ? I$;Z$;"SORRY—NONE LEFT.":? :GOTO 3100
5000 REM INITIALIZE
5010 REM ** START PROLOGUE **
5020 OPEN #1,4,0,"K:":DIM AZ$(100),BL$(37),I$(3),WHO$(4):? CHR$(125):POSITION 2,4:I$ = "  "
5030 POKE 752,1:BL$ = "                                    ":? I$;REM 37 SPACES
5040 ? "THE CAVERNS OF ENILT FOS"
5050 ? :?
5060 ? I$;"YOUR FIGHTER, ON THIS TRIP, IS":?
5070 ? I$;"JOE, A MALE HUMAN OF AVERAGE":?
5080 ? I$;"SIZE AND CAPABILITIES":? :?
5090 REM CONTINUED AT 5500
5100 REM ** VARIABLES **
5110 DOING = 0
5130 EP = INT(6*RND(5)) + 2
5140 HP = INT(4*RND(5)) + 1
5150 EQ = EP:HQ = HP
5160 SAFE = 3
5170 TIME = 0
5180 WALL = 0
5200 REM ** STRINGS **
5210 DIM NAME$(3),PROF$(1),Z$(1),E$(18),P$(21),ST$(37),W$(22):NAME$ = "JOE":PROF$ = "F"
5230 Z$ = CHR$(253):REM ***** BUZZER
5240 E$ = "SOMEONE IS COMING!"
5260 P$ = "WHAT WILL YOU DO NOW?"
5270 ST$ = "NAME  PR HEALTH ENERGY  HEAL ENERGZ"
5280 W$ = "YOU ARE FACING A WALL."
5300 REM ** STRING ARRAYS **
5310 DIM C$(98)
5320 C$ = "YOU ARE LOOKING DOWN A CORRIDOR.,A CORRIDOR STRETCHES BEFORE YOU.,YOU ARE
     FACING INTO A CORRIDOR.,"
5350 DIM O$(87)
5360 O$ = "F/SP:FORWARD R/- >:RIGHT L/< -:LEFT,F:FIGHT R:RUN,1 - 9:FIGHT H:HEAL E:ENERGIZE,"
5400 REM ** COMBAT SPECS **
5410 DIM HEALTH(1):DIM ENERGY(1):DIM VIT(1):DIM IN(1)
5420 DIM AC(1):DIM AI(1):DIM DC(1):DIM DI(1):DIM HIT(1)
5450 HEALTH(1) = 100:AC(1) = 100
5460 ENERGY(1) = 100:DC(1) = 100
5500 REM ** MORE PROLOGUE **
5510 DIM Q$(35):GOSUB 1220:? I$;
5520 ? "      - - COMBAT CHART - -"
5540 READ Q$:? :FOR X = 1 TO 8
5550 READ Q$:? :? Q$
5560 GOSUB 1230:NEXT X:?
5570 DATA,     ATTACK: NONE  MODERATE ALL-OUT,D,E  SLIGHT     1     4     7,F
5575 DATA E  MEDIUM     2     5     8,N,S  STRONG     3     6     9,E
5580 ? I$;"   (ANY KEY TO CONTINUE)":? :?
5590 POKE 752,0:POSITION 18,23:? " ";:GET #1,Q:POKE 752,1:?
5600 REM ** INTRODUCTION **
5610 PRINT
5620 FOR X = 1 TO 6
5630 READ Q$:? I$;Q$
5640 GOSUB 1220:? :NEXT X
5660 DATA JOE - YOU ARE IN A CORRIDOR.,IT STRETCHES AS FAR AS YOU CAN,SEE IN BOTH
     DIRECTIONS - -
5661 DATA DIMLY LIT BY AN OCCASIONAL TORCH,IN A WALL HOLDER.  THE NEXT,MOVE IS UP TO YOU.
5700 REM ** COMBAT STRINGS **
5710 DIM HIT$(30),DAM$(58)
5720 HIT$ = "YOU HIT HIM!.HE HIT YOU!,"
5740 DAM$ = "MISSED.,ONLY A SCRATCH.,MODERATE DAMAGE.,SERIOUS DAMAGE!,"
5770 DIM LT$(31)
5780 LT$ = "ENERGY PILLS!,HEALING POTIONS!,"
5800 GOTO 110
```

Listing 2

```
3 REM
4 REM THE DUNGEON MACHINE
5 REM PART 2: MOVEMENT
6 REM
7 REM V: 108
8 REM
74 IF AZ$(IX,IX) < > "/" THEN 84
100 REM + + + EACH ROUND + + +
```

```
110 REM ** EACH PAGE **
140 FOR X = 1 TO CV - 18
300 REM ** PROMPT & INPUT **
320 POSITION 2,20:GOSUB 66
330 ? P$;:CH = PEEK(85)
370 POSITION CH + 1,20:POKE 752,0:? " ";
460 GOTO 4000
500 REM DELETE 500 THROUGH 730
935 HS = INT(3*RND(7)):VS = INT(3*RND(7))
940 DOING = 0:GOTO 4500
1310 POSITION CH + 2,20:? CHR$(Q)
1330 GOSUB 62:POSITION CH + 2,20:? " ";:GOTO 200
1350 GOSUB 1240:POSITION 2,22
1400 REM ** ADVANCE VWPT **
1410 HL = HS:VL = VS
1420 IF D = 0 THEN VL = VL + 1
1430 IF D = 1 THEN HL = HL + 1
1440 IF D = 2 THEN VL = VL - 1
1450 IF D = 3 THEN HL = HL - 1
1470 RETURN
4000 REM + + + MOVEMENT + + +
4010 REM ** DISTRIBUTE **
4020 SP = 0
4030 IF Q = 76 THEN 4300:REM LOOK
4040 IF Q = 30 OR Q = 31 OR Q = 42 OR Q = 43 THEN 4400:REM TURN
4050 IF Q = 68 THEN 4200:REM ROOM DESCRIPTION
4060 IF Q = 67 THEN 4800:REM COMPASS
4070 REM R)OOM DESCR. Z)OTHER O)PEN DOOR
4080 IF Q<>32 THEN 1300
4100 REM ** FORWARD **
4110 IF FO = 1 OR FO = 3 THEN 1300:REM CAN'T DO THAT
4120 IF FO = 2 THEN RD = 0:REM NEW ROOM, DESCRIBE IT
4130 GOTO 1400
4140 HS = HL:VS = VL
4150 A = 1
4160 IF RD THEN 4500
4200 REM ** ROOM DESCRIPTION **
4220 FOR X = 1 TO 4
4230 AZ$ = RM$:IZ = 5*SQ(12*VS + 4*HS) + X:GOSUB 70:PRINT I$;AZ$
4240 NEXT X
4250 PRINT :RD = 1
4260 GOTO 4500
4300 REM ** LOOK AHEAD **
4310 IF FO = 1 THEN 1300
4320 IF FO = 3 THEN 1300
4330 GOSUB 1400
4340 A = 0:GOSUB 4520
4350 POKE 4520
4400 REM ** TURN **
4405 A = 1
4410 IF Q = 30 OR Q = 43 THEN 4450:REM (LEFT TURN)
4420 D = D + 1:IF D>3 THEN D = D - 4
4430 GOTO 4500
4450 D = D - 1:IF D<0 THEN D = D + 4
4500 REM ** WHERE YOU ARE **
4510 HL = HS:VL = VS
4520 VX = VL + 1:IF VX = 3 THEN VX = 0
4530 HX = HL + 1:IF HX = 3 THEN HX = 0
4540 WT(0) = SQ(12*VX + 4*HL + 1)
4545 WT(1) = SQ(12*VL + 4*HX + 2)
4550 WT(2) = SQ(12*VL + 4*HL + 1)
4555 WT(3) = SQ(12*VL + 4*HL + 2)
4560 HX = D - 1:IF HX = - 1 THEN HX = 3
4565 LT = WT(HX)
4570 HX = D + 1:IF HX = 4 THEN HX = 0
4575 RT = WT(HX)
4580 FT = WT(D)
4585 IF A = 1 THEN FO = FT
4600 REM ** WHAT YOU SEE **
4610 AZ$ = VT$:IZ = A:GOSUB 70:PRINT I$;AZ$;:AZ$ = VW$:IZ = FT:GOSUB 70:PRINT AZ$;
4620 AZ$ = VZ$:IZ = FT<2:GOSUB 70:PRINT AZ$
4630 IF LT<>FT THEN 4670
```

```
4640 IF RT<>LT THEN 4660
4650 ? I$;" AND ON EITHER SIDE.":GOTO 100
4660 IF LT=0 THEN ? I$;" THE SAME ON YOUR LEFT, AND":GOTO 4680
4665 ? I$;" ANOTHER ON YOUR LEFT, AND":GOTO 4680
4670 AZ$=VW$:IZ=LT:GOSUB 70:? I$;" ";AZ$;" ON YOUR LEFT, AND"
4680 AZ$=VW$:IZ=RT:GOSUB 70:? I$;" ";AZ$;" TO THE RIGHT."
4690 GOTO 800
4800 REM ** COMPASS **
4830 ? I$:"FACING ";
4840 GOSUB 1230:AZ$=DIR$:IZ=D:GOSUB 70:? AZ$;
4850 ? " V:";VS;" H:";HS;
4860 ? " SQ:";SQ(12*VS+4*HS+3)
4870 GOSUB 1220:GOTO 800
4890 REM
4900 REM
5020 OPEN #1,4,0,"K:":DIM AZ$(356),BL$(37),I$(3),WHO$(4):? CHR$(125):POSITION 2,4:I$=" "
5300 REM ** STRING ARRAYS**
5320 C$="YOU ARE LOOKING DOWN A CORRIDOR./A CORRIDOR STRETCHES BEFORE YOU./YOU ARE
     FACING INTO A CORRIDOR./"
5350 DIM O$(124)
5360 O$="SP:FORWD  <-:LT  ->:RT  L:LOOK AHEAD     D:ROOM DESCR  C:COMPASS/F:FIGHT  R:RUN/"
5380 O$(93)="1-9:FIGHT  H:HEAL  E:ENERGIZE/"
5600 REM ** INTRODUCTION **
5660 DATA JOE - YOU ARE IN AN ISOLATED,GROUP OF ROOMS - SOMEWHERE,UNDERGROUND. THE
     WALLS ARE
5661 DATA OF NATIVE STONE - REINFORCED,WITH AGE-BLACKENED TIMBERS.,IT'S CHILLY DOWN HERE!
5670 GOSUB 1210
5700 REM ** COMBAT STRINGS **"
5720 HIT$="YOU HIT HIM! / HE HIT YOU! /"
5740 DAM$="MISSED./ONLY A SCRATCH./MODERATE DAMAGE.//SERIOUS DAMAGE!/"
5780 LT$="ENERGY PILLS!/HEALING POTIONS!/"
5800 REM **   MOVEMENT STRINGS **
5810 DIM DIR$(22):DIM VT$(19)
5820 DIM VW$(43):DIM VZ$(28)
5830 VT$="YOU SEE /THERE IS /"
5840 DIR$="NORTH/EAST/SOUTH/WEST/"
5850 VW$="NOTHING/A WALL/AN OPEN DOOR/A CLOSED DOOR/"
5880 VZ$=" AHEAD, / IN FRONT OF YOU, /"
5890 DIM WT(3)
5990 REM
6000 REM + + + THE ROOMS + + +
6100 REM ** PARAMETERS **
6110 DIM SQ(35):REM 3 X 3, WITH 4 FILES - APPLE VERSION = SQ(2,2,3)
6120 DIM RM$(356):REM 4 ROOMS
6200 REM ** ROOM NUMBERS **
6210 FOR I=0 TO 35 STEP 4
6220 READ RZ:SQ(I)=RZ:NEXT I
6240 DATA 1,1,2,3,4,4,3,4,4
6300 REM ** WALLS **
6310 FOR I=1 TO 35 STEP 4
6320 READ RZ:SQ(I)=RZ:NEXT I
6330 DATA 1,1,1,2,2,1,0,0,0
6350 FOR I=2 TO 35 STEP 4
6360 READ RZ:SQ(I)=RZ:NEXT I
6370 DATA 1,0,2,1,1,0,1,1,0
6400 REM **   SPECIAL SQUARES **
6410 FOR I=3 TO 35 STEP 4
6420 READ RZ:SQ(I)=RZ:NEXT I
6430 DATA 0,1,2,3,4,5,6,7,8
6500 REM ** ROOM DESCRIPTIONS **
6530 RM$="//////YOU ARE IN A SHORT HALLWAY,/TWENTY FEET LONG BY TEN WIDE/THERE IS A DOOR AT
     THE END AND/"
6531 RM$(95)="TWO DOORS ON ONE SIDE./"
6540 RM$(118)="/A BOX OF A ROOM, TEN/FEET SQUARE.  THERE IS/ONLY ONE DOOR AND
     NO/FURNITURE./"
6550 RM$(195)="/ANOTHER SHORT HALL, SHAPED LIKE/THE LAST ONE, BUT AT A RIGHT/ANGLE TO IT.
     THERE IS A DOOR/"
6551 RM$(286)="AT ONE END./"
6560 RM$(298)="/A LARGE ROOM,/TWENTY FEET SQUARE./THERE IS ONLY ONE/DOOR./"
6970 D=1:REM FACING EAST
6980 A=1:REM 'THERE IS'
6990 GOTO 4200
```